McCormick

Northwestern Engineering

Department of Electrical Engineering and Computer Science

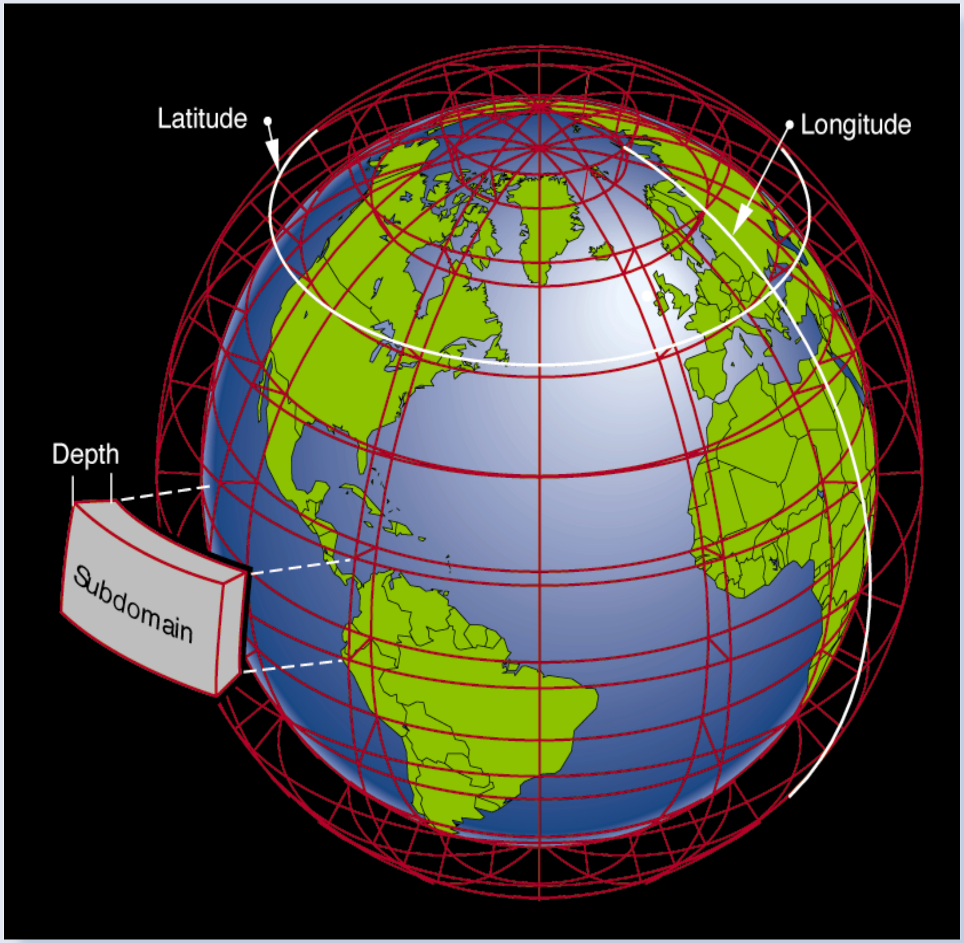
Next Generation I/O Panel (Are we Addressing the Right Problem? Think Purpose!)

Alok Choudhary

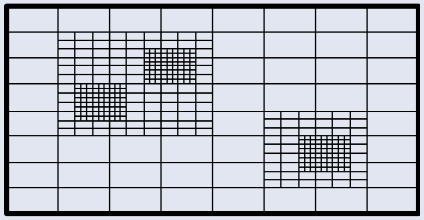
HEC FSIO 2011

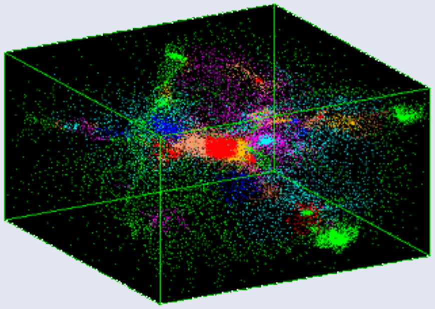
LET'S LOOK AT THE USERS- WHAT DOES A USER WANT/ NEED?

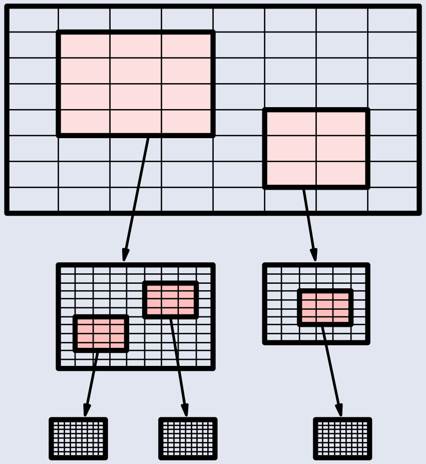
How Do I Represent and Manage My Data? Not How File System Works and How do I manage millions of files

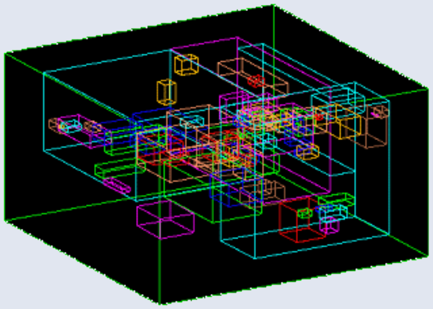


How Do I discover Knowledge from Complex Data Sets? Not How do I Optimize Reading Millions of Files and from millions of processors?

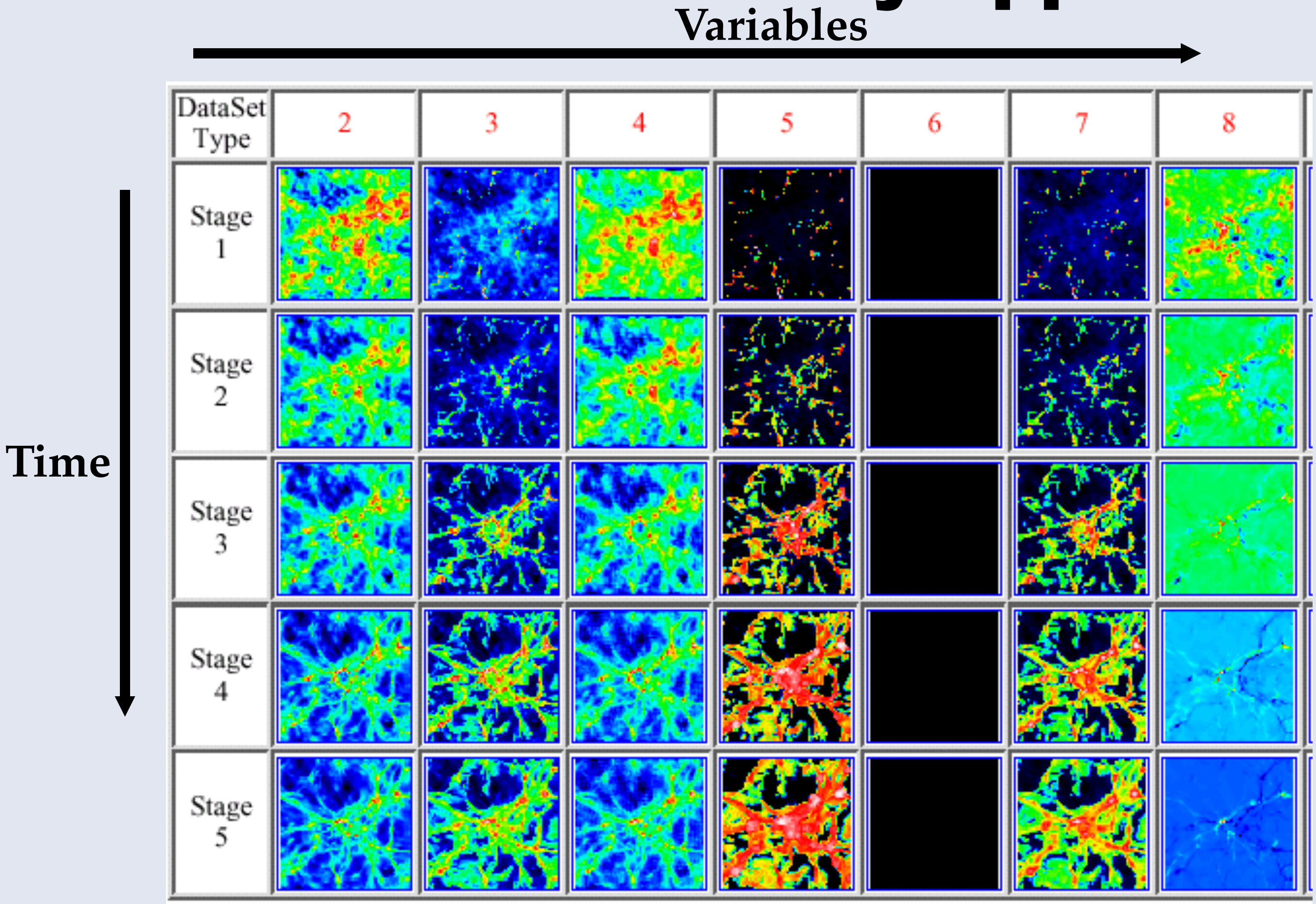






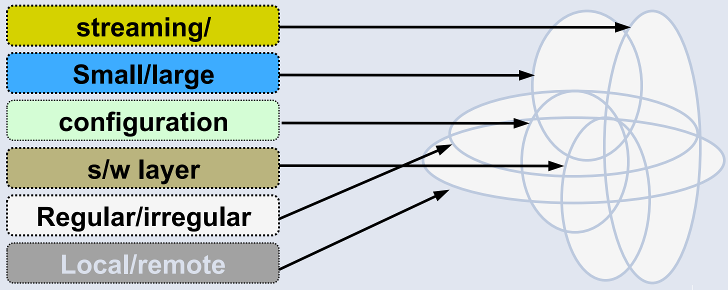


How Do I perform queries in a manner which relates to my application?



What Does a User Get?

User specifies how



streaming/

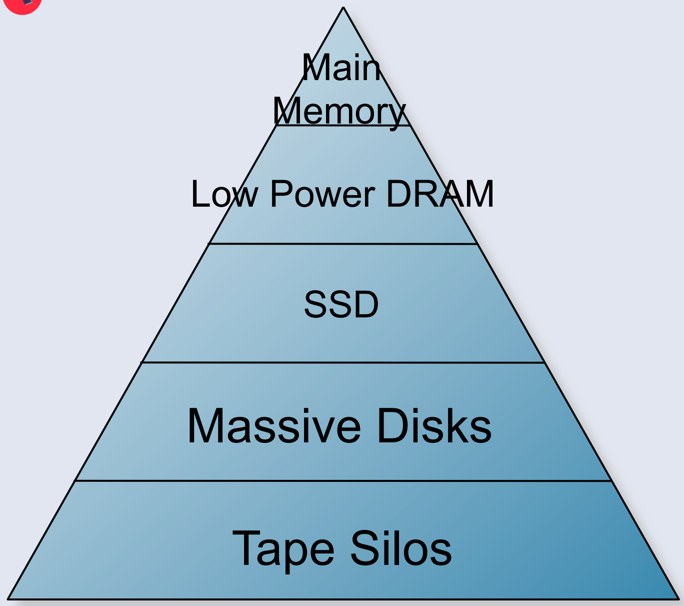
Small/large

configuration

s/w layer

Regular/irregular

Local/remote



Main Memory

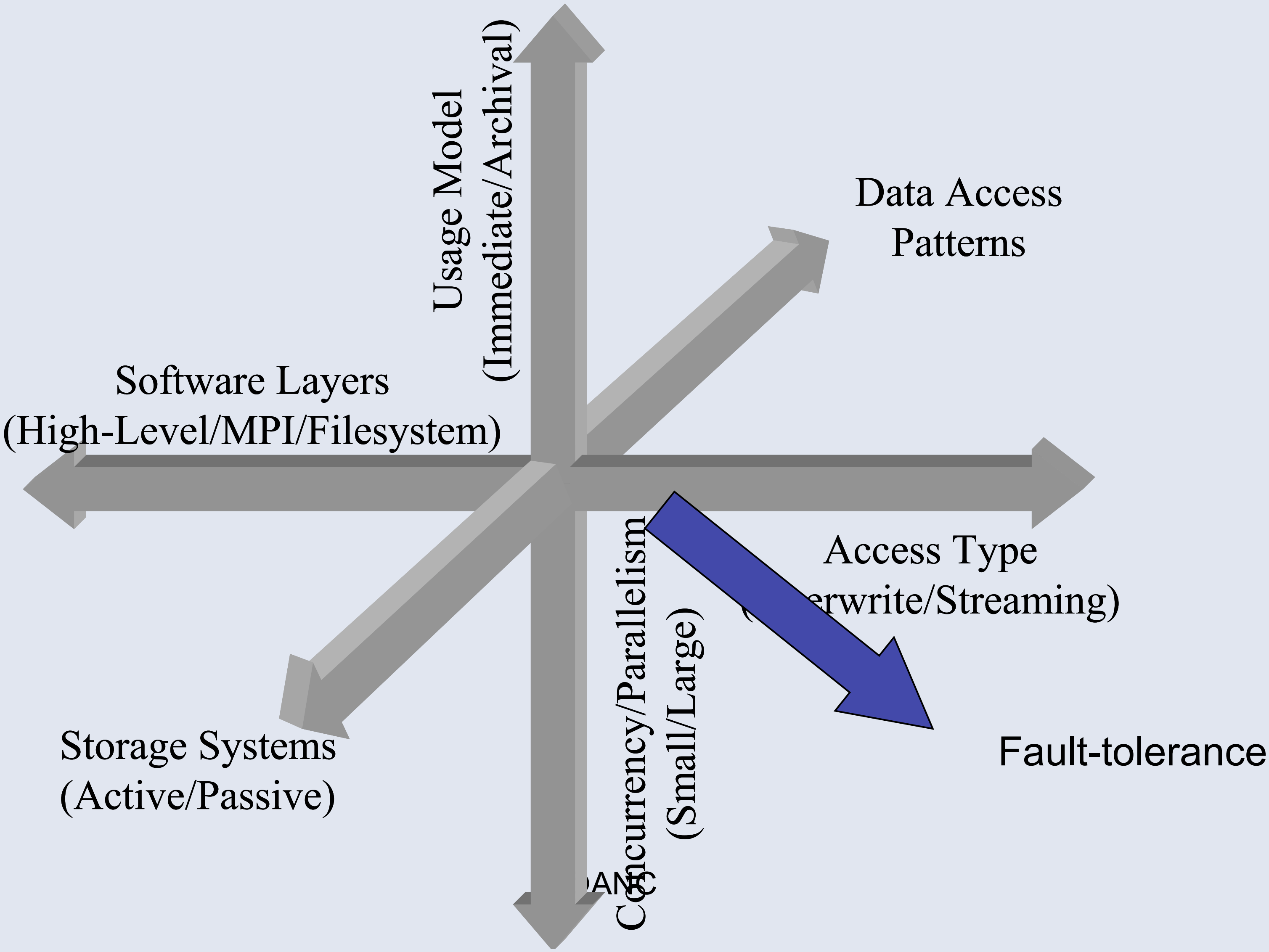
Low Power DRAM

SSD

Massive Disks

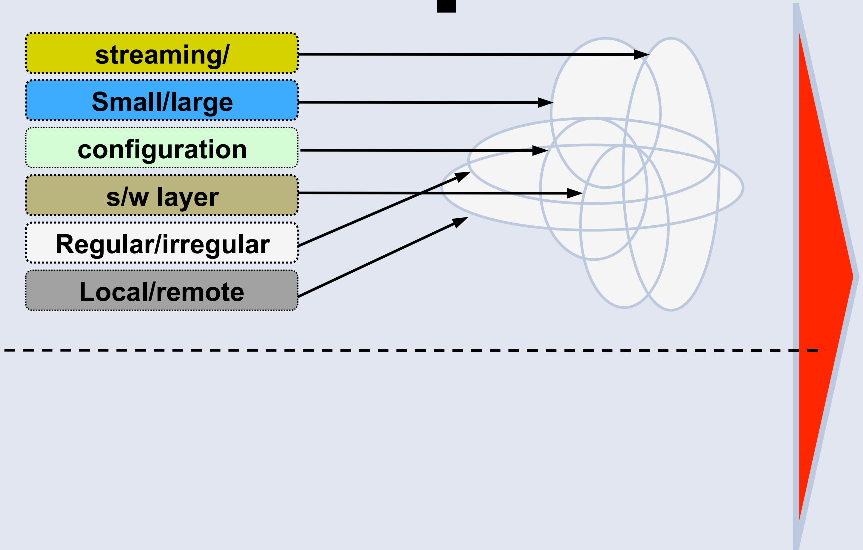
Tape Silos

Complexity


Usage Model
(Immediate/Archival)
Data Access
Patterns
Software Layers
(High-Level/MPI/Filesystem)
Access Type
(erwrite/Streaming)
Fault-tolerance
Storage Systems
(Active/Passive)
Concurrency/Parallelism
(Small/Large)
ANC

Current   Goal

Decouple “What” from “How”



streaming/

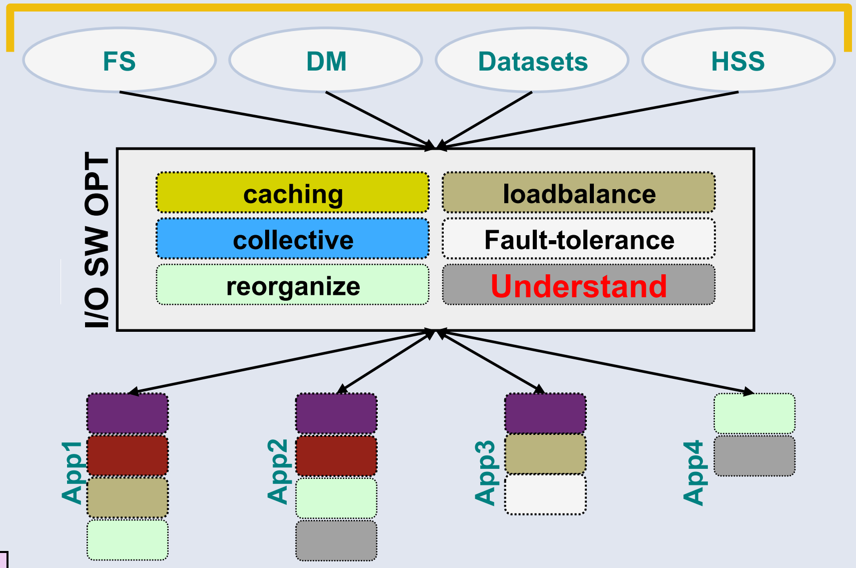
Small/large

configuration

s/w layer

Regular/irregular

Local/remote



FS

DM

Datasets

HSS

I/O SW OPT

caching

loadbalance

collective

Fault-tolerance

reorganize

Understand

App1

App2

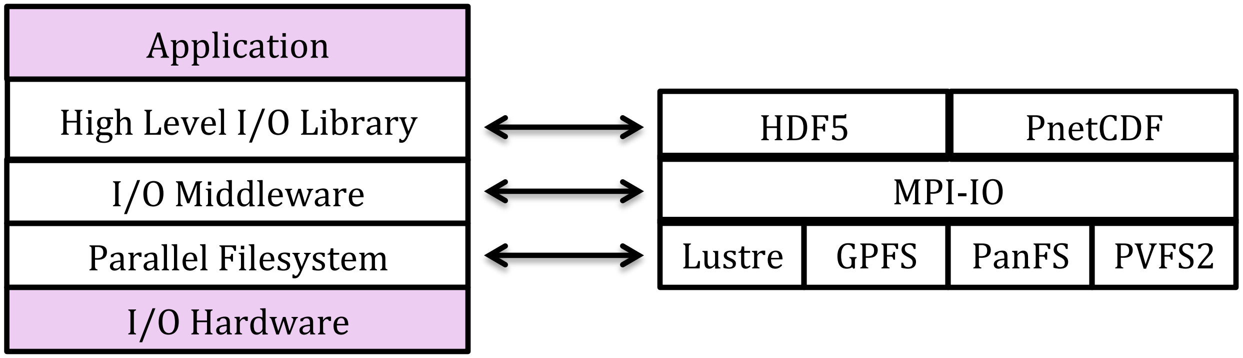
App3

App4

- Is there a way to specify high-level information?

- Proactive
- Performance
- Portability

I/O Software Stack for Scientific Computing



Application

High Level I/O Library ↔ HDF5 | PnetCDF

I/O Middleware ↔ MPI-IO

Parallel Filesystem ↔ Lustre | GPFS | PanFS | PVFS2

I/O Hardware

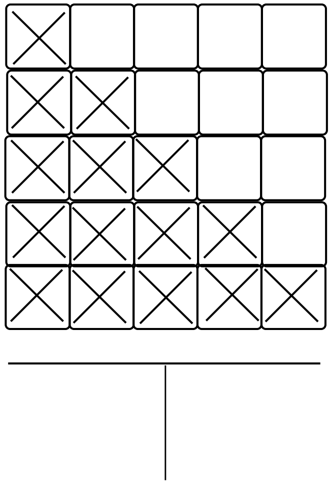
High Level I/O Library

- Storage data models developed in the 1990s; Network Common Data Format (netCDF) and Hierarchical Data Format (HDF)
- Multidimensional array based data models
- A portable, self-describing on-disk file format
- HDF5
  - Supports regular grid based data models

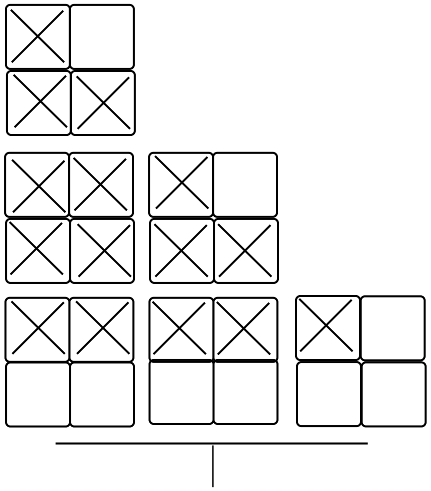
Parallel netCDF

- A parallel I/O library based on original netCDF
- Data Model:
  - Collection of variables in single file
  - Typed, multidimensional array variables
  - Attributes on file and variables
- Features:
  - Portable data format
  - Noncontiguous I/O in memory using MPI datatypes
  - Noncontiguous I/O in file using sub-arrays
  - Collective I/O

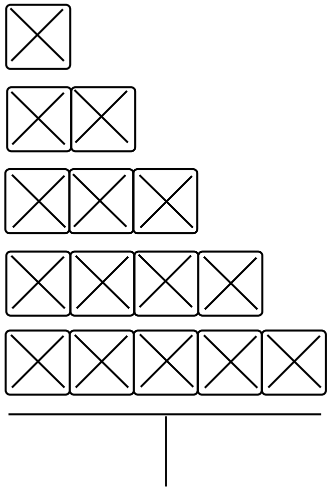
Example: Lower Triangular Matrix



netCDF: fixed dimensions



HDF5: Potential for odd interactions between application data layout and chunk allocation



Lower-triangular aware storage model and layout

Some Observations

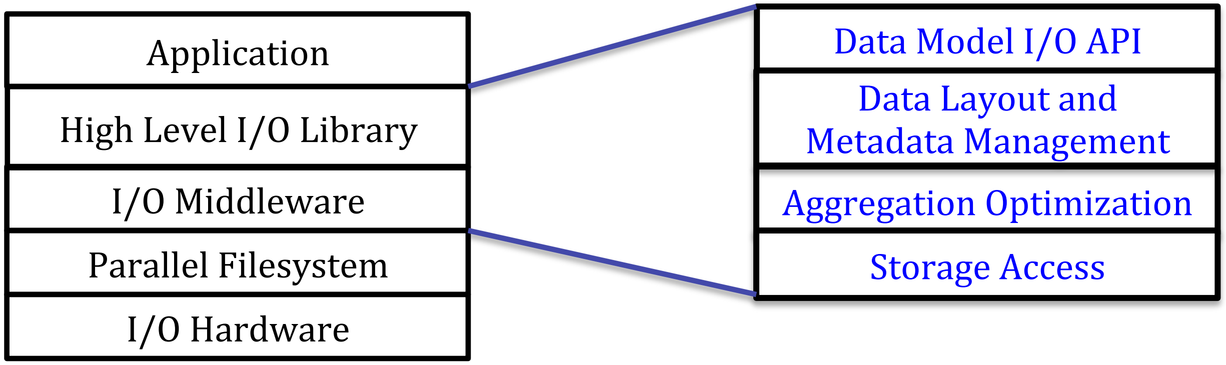
- I/O library interfaces still based on low-level vectors of variables
- Lack of support for sophisticated data models, e.g. AMR, unstructured Grids, Geodesic grid, etc
- Gap between application data model and I/O library data model
- Require too much work at application level to achieve close to peak I/O performance

DAMSEL Goals

- Provide higher-level data model API to describe more sophisticated data models
- Enable exascale computational science applications to interact conveniently and efficiently with storage through the data model API
- Develop a data model storage library to support these data models, provide efficient storage data layouts

DAMSEL – A Data Model Storage Library

- A set of data models I/O APIs relevant to computational science applications
- A data layout component that maps these data models onto storage efficiently
- A rich metadata representation and management layer to handle both internal metadata and that generated by users and external tools
- I/O optimizations: adaptive collective I/O, request aggregation, and virtual filing



Application

High Level I/O Library

I/O Middleware

Parallel Filesystem

I/O Hardware

Data Model I/O API

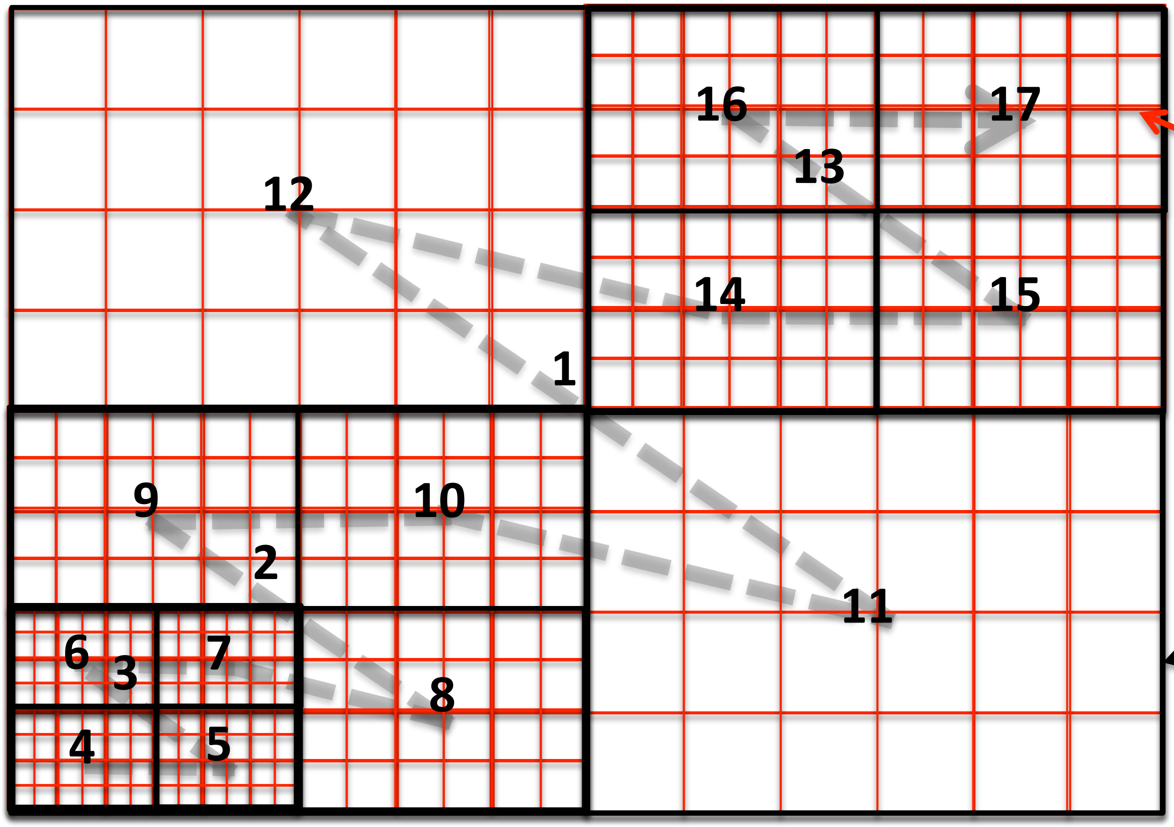
Data Layout and Metadata Management

Aggregation Optimization

Storage Access

DAMSEL for FLASH

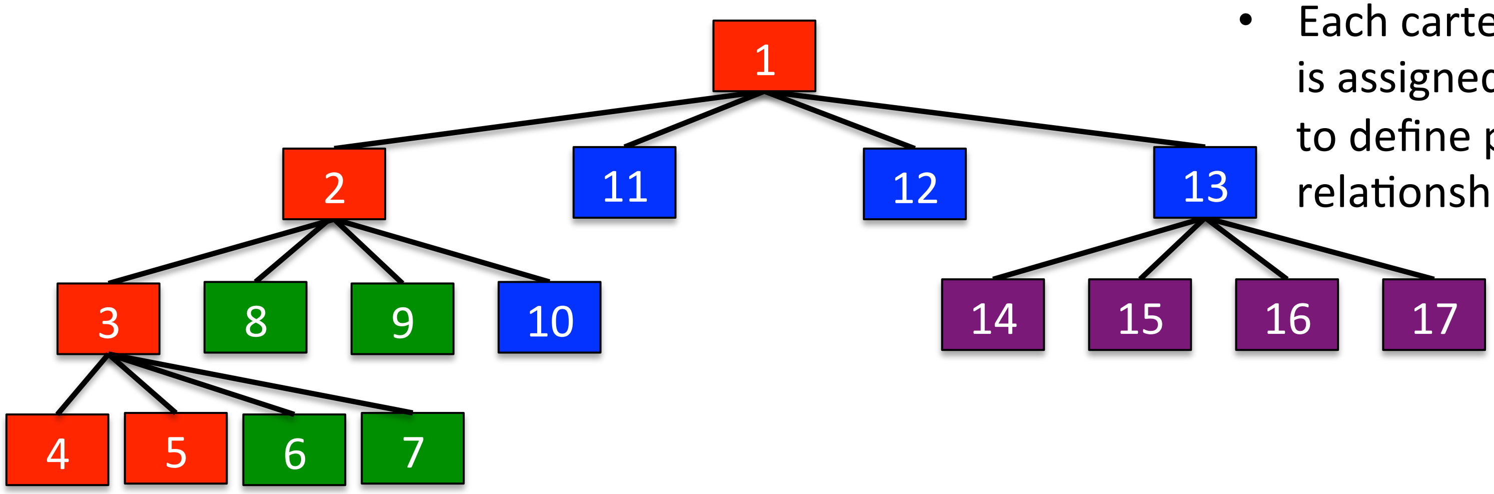
- Goal: to describe hierarchical/structural and solution information through API
- Entity
  - FLASH cell as rectangle entity in DAMSEL
  - FLASH Block as Cartesian Mesh entity in DAMSEL
- Entity Sets
  - FLASH blocks assigned to entity sets to define hierarchical/ structural information
- Tags
  - Only for solution data



Solution data on cells as **tags** in DAMSEL

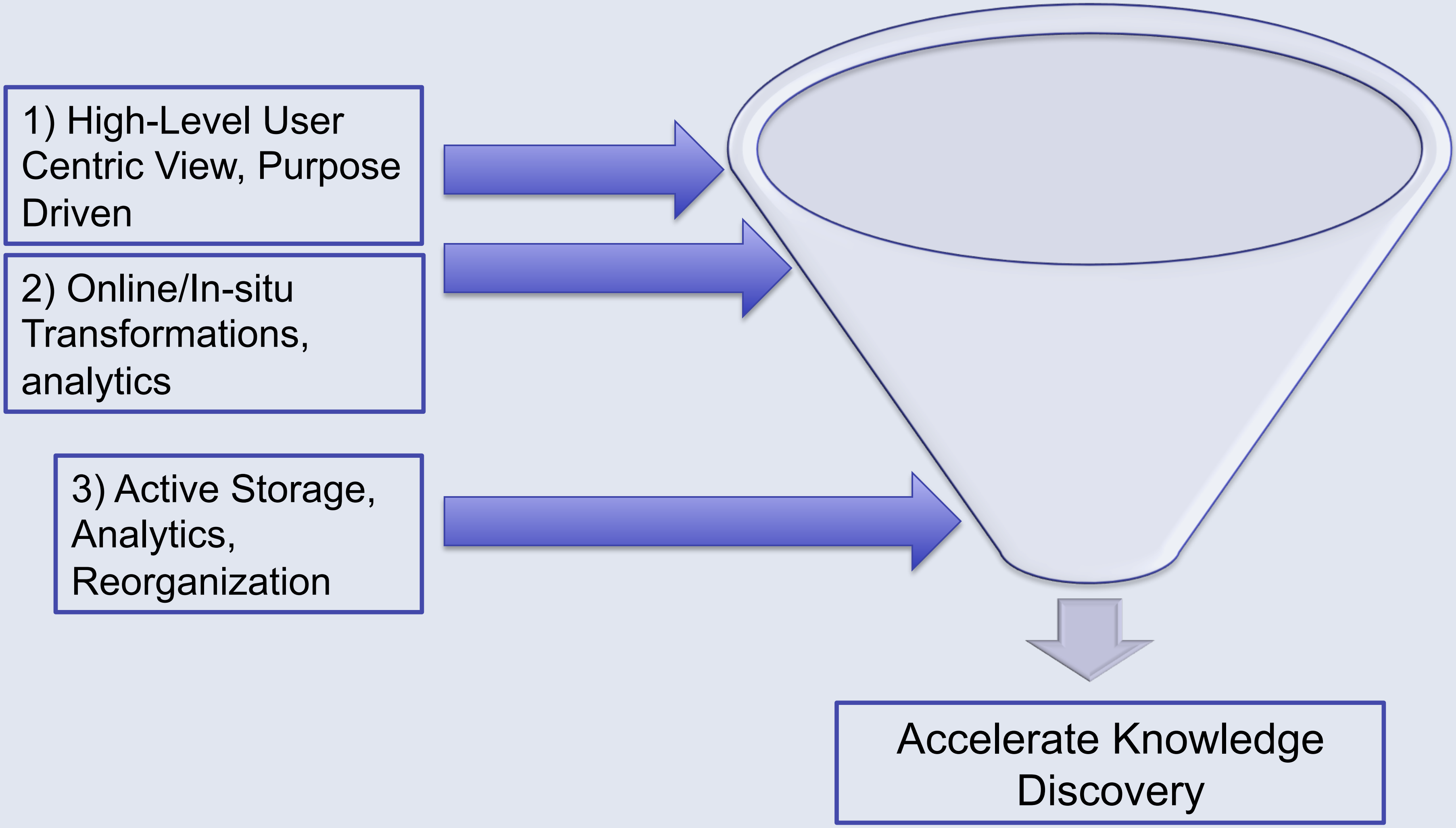
- FLASH cells as **rectangle entity** in DAMSEL
- FLASH blocks as **cartesian mesh entity** in DAMSEL

Morton order



- Each cartesian mesh entity is assigned to an **entity set** to define parent/child relationship in DAMSEL

Next Generation I/O - Three Ideas:



1) High-Level User Centric View, Purpose Driven

2) Online/In-situ Transformations, analytics

3) Active Storage, Analytics, Reorganization

Accelerate Knowledge Discovery

Summary

- Think Processor Evolution when thinking I/O
  - Desktop, laptop, Mainframe, embedded, mobile, graphics, games, etc etc… Designed to solve a problem
- Think application and user model and needs and not how to make things work with file system
- Working in user's language will accelerate knowledge discovery – and they will come
- Ultimately, the purpose of I/O should drive architecture at all levels